Scalable Web Application with Load Balancing on Azure

This project demonstrates how to deploy a highly available and scalable web application on Microsoft Azure using Terraform. The deployment includes a Virtual Machine Scale Set (VMSS), an Azure Load Balancer, auto-scaling rules, and monitoring with Azure Monitor.

Features

- **Scalability**: Automatically scale up or down based on resource usage (e.g., CPU or memory).
- **Load Balancing**: Distribute incoming traffic evenly across virtual machines using Azure Load Balancer.
- **Monitoring and Alerts**: Track performance metrics with Azure Monitor and receive alerts for defined thresholds.
- **Infrastructure as Code**: Use Terraform for consistent, reproducible deployments.

Prerequisites

- Azure account with appropriate permissions.
- Terraform installed on your local machine.
- Azure CLI installed and authenticated.

Architecture Overview

1. **Virtual Machine Scale Set (VMSS)**: Hosts the web application instances.
2. **Azure Load Balancer**: Distributes incoming traffic across VMSS instances.
3. **Auto-scaling Rules**: Dynamically adjust the number of instances based on CPU or memory usage.
4. **Azure Monitor**: Provides metrics and alerts for system performance.
5. **Azure Storage**: Stores web application resources.

Deployment Steps

1. Clone the Repository

```
git clone <repository-url>
cd <repository-folder>
```

2. Configure Variables

Edit the `variables.tf` file to match your Azure environment:

```
variable "resource_group_name" {
  default = "my-resource-group"
}

variable "location" {
```

```
  default = "East US"
}

variable "vm_size" {
  default = "Standard_B1ms"
}

# Add other variables as needed
```

Alternatively, create a `terraform.tfvars` file to override default values:

```
resource_group_name = "custom-resource-group"
location            = "West Europe"
```

3. Initialize Terraform

```
terraform init
```

This command downloads necessary provider plugins and initializes the project.

4. Plan the Deployment

```
terraform plan
```

Review the execution plan to ensure it matches your expectations.

5. Apply the Deployment

```
terraform apply
```

Type `yes` to confirm the changes and deploy the infrastructure.

6. Verify the Deployment

- Access the web application using the public IP address of the load balancer.
- Check the Azure portal to verify the VMSS, load balancer, and other resources are created.

7. Monitoring and Alerts

- Use Azure Monitor to track metrics (e.g., CPU usage) for VMSS instances.
- Configure alerts in the Azure portal to notify you of performance issues.

Directory Structure

```
|-- main.tf              # Main Terraform configuration file
|-- variables.tf         # Input variable definitions
|-- outputs.tf           # Outputs for the deployment
|-- terraform.tfvars     # Optional file for variable overrides
|-- modules/             # Optional directory for reusable Terraform modules
```

Auto-scaling Configuration

Auto-scaling rules are defined in the `main.tf` file. Example:

```
resource "azurerm_monitor_autoscale_setting" "example" {
  name                = "example-autoscale"
  resource_group_name = azurerm_resource_group.example.name
  location            = azurerm_resource_group.example.location

  profile {
    name = "default"

    capacity {
      minimum = 1
      maximum = 5
      default = 2
    }

    rule {
      metric_trigger {
        metric_name        = "Percentage CPU"
        metric_resource_id = azurerm_virtual_machine_scale_set.example.id
        operator           = "GreaterThan"
        statistic          = "Average"
        threshold          = 75
        time_grain         = "PT1M"
        time_window        = "PT5M"
        time_aggregation   = "Average"
      }

      scale_action {
        direction = "Increase"
        type      = "ChangeCount"
        value     = "1"
        cooldown  = "PT1M"
      }
    }
  }
}
```

Clean-Up

To remove the deployed resources:

```
terraform destroy
```

Type `yes` to confirm and delete the infrastructure.

Notes

- Ensure your Azure account has sufficient quotas for the requested resources.
- Follow best practices for securing your Terraform state file, especially if using remote state storage.

Contributions

Feel free to submit issues or pull requests to improve this project.

License

This project is licensed under the MIT License. See the `LICENSE` file for details.